AFRL-IF-RS-TR-2002-64
Final Technical Report
April 2002

DYNAMIC, COOPERATING BOUNDARY CONTROLLERS

Boeing Defense & Space Group

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. E295

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** Apr 02	**3. REPORT TYPE AND DATES COVERED** Final Sep 96 - Feb 98

4. TITLE AND SUBTITLE DYNAMIC, COOPERATING BOUNDARY CONTROLLERS	**5. FUNDING NUMBERS** C - F30602-96-C-0318 PE - 62301E PR - E017 TA - 01 WU - 06
6. AUTHOR(S) Dan Schnackenburg	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Boeing Defense & Space Group Research & Technology PO Box 3999 Seattle, WA 98124-2499	**8. PERFORMING ORGANIZATION REPORT NUMBER** D658-10822-1
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency 3701 North Fairfax Drive Arlington, VA 22203-1714 — AFRL/IFGB 525 Brooks Road Rome, NY 13441-4505	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2002-64

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: John Feldman, IFGB, 315-330-2664

12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

This project has prototyped a preliminary design concept for a capability, enabling networks of networks to cooperate in the detection of system attacks, learn about the attack behavior, and dynamically reconfigure to protect the greater network infrastructure. This can be enabled by providing a mechanism for intrusion detection systems and boundary controllers to coordinate their actions. The focus of this concept has been to develop, implement and demonstrate an Intruder Detection and Isolation Protocol (IDIP) which can be used to track network intruders to their point of entry within the "cooperating" network of networks, thus enabling network-level access control policies to be dynamically changed in response to the detected attacks.

14. SUBJECT TERMS Intrusion Detection, Boundary Controllers, Automated Response, Intrusion Isolation, Dynamic Response	**15. NUMBER OF PAGES** 36
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

LIST OF FIGURES

Dynamic, Cooperating Boundary Controllers

1 INTRODUCTION

This objective of this effort was to prototype a capability, enabling networks of networks to cooperate in the detection of system attacks, mitigating their effects and dynamically reconfiguring to protect the greater network infrastructure from further attacks launched via the offending source. The solution sought had to satisfy the following design criteria:

1. Scalability from small to very large systems.
2. Robustness.
3. End-user transparency.
4. Simplicity.
5. Protection against spoofing.
6. Compatibility with multiple encryption technologies.

The approach taken was to develop the required technology including, a concept of operations, specific mechanisms, and a special protocol suitable for achieving the objective. The concept of operations was then proved by the development of separate implementations of that protocol for three representative boundary control/intrusion detection examples, and a feasibility demonstration of the resultant technology's.

A new intrusion response protocol and its concept of operations were designed to enable the following functions--

1. Tracking down the intra-network launch point across a network of networks (given that the intruder had successfully compromised a host within the network of networks).
2. Dynamically altering the filtering rules of firewalls and filtering routers based on suspicious behavior identified at remote devices.
3. Dynamically altering the monitoring rules of access control devices and intrusion detection systems based on suspicious behavior identified at remote devices.
4. Synthesizing the intrusion detection capabilities of both access control and intrusion detection system to gain a better picture at the system level of network intrusions, thus providing the underlying mechanisms to enable "detection fusion."
5. Providing for centralized monitoring of progress in accomplishing the tracking down function.
6. Maintaining autonomy of sub-networks while cooperating with neighboring security devices.

1.1 Background

Government, and commercial information systems are vulnerable to attacks., System administrators typically use statically configured "boundary controller" components (e.g., filtering routers and firewalls) at the network boundaries to protect their systems from these onslaughts. These boundary controllers typically operate as standalone security barriers protecting their internal system domain against **outside** intruders. Because these boundary controllers concentrate on protecting the internal domain from outside attacks, the mechanisms tend to provide limited protection from **internal** user activity. Therefore, once an attacker succeeds in compromising a host inside a protected domain, the attacker often can operate with relative impunity inside the networked system launching attacks against other hosts inside the network of networks. These attacks frequently are very effective because within a networked system, sub-networks typically are not protected from each other. Attackers who get past the external system barriers have greater opportunities to use the compromised network to penetrate additional subsystems and even use local resources to launch attacks on critical systems. One method of changing this external attack into an apparently internal legitimate user would be to deposit a Trojan horse that continues the attack for the penetrator. Placing Trojan horses within a network is relatively simple. Intruders can use ActiveX applets, Java applets, and document macros to install Trojan horses within internal system components, circumventing typical firewall controls. For example , the penetrator can use services (e.g. web access and email) that must pass through the firewall in most organizations. Both web access and email services allow a remote user to place executable code on the destination machine, and this code can then be used to launch attacks across the organization's Intranet.

To help protect their networks, system administrators can deploy intrusion detection systems throughout the enterprise to determine when outside penetrators or malicious insiders are attacking critical components. Traditionally, intrusion detection components and network boundary controllers (e.g. firewalls and filtering routers) have operated independently in providing system protection. Detected anomalies are typically reported to system administrators by the intrusion detection components, but by the time a system administrator determines and executes the correct response to the intrusion, the attack has more often then not succeeded.

For the enterprise to reasonably respond to network-based attacks, mechanisms for cooperation among boundary controllers and intrusion detection system must be in place. Although more commercial off-the-shelf (COTS) intrusion detection systems have added real-time response mechanisms that can change local boundary controller policy to block incoming data streams based on the detected anomaly's source and target addresses and targeted services to their functionality. There are two difficulties inherent with these solutions. They are that (1) the response can only occur with boundary controllers known to the intrusion detection system, and (2) the intrusion detection system must know how to configure each boundary controller that it needs to control. Figure 1 shows a typical configuration for these intrusion detection systems. As seen in Figure 1, if the attacker were spoofing Host *A*'s source address, then the intrusion detection system directed response to the firewall would not block the attacker, but would block legitimate traffic from Host *A*. This would allow the attacker to use the response mechanism as a denial of service attack.

To minimize the impact on the rest of the networked system, a mechanism is required that (1) enables the response to be closer to the attacker (minimizing impact on legitimate traffic; and (2) allows interoperability among a wider variety of intrusion detection systems. The Intruder Detection and Isolation Protocol (IDIP) provides this interoperability and also provides boundary controllers in the path of the attack with sufficient information to permit tracking the intruder to the network component from which the attack is being launched and to block the attack traffic closer to the source.

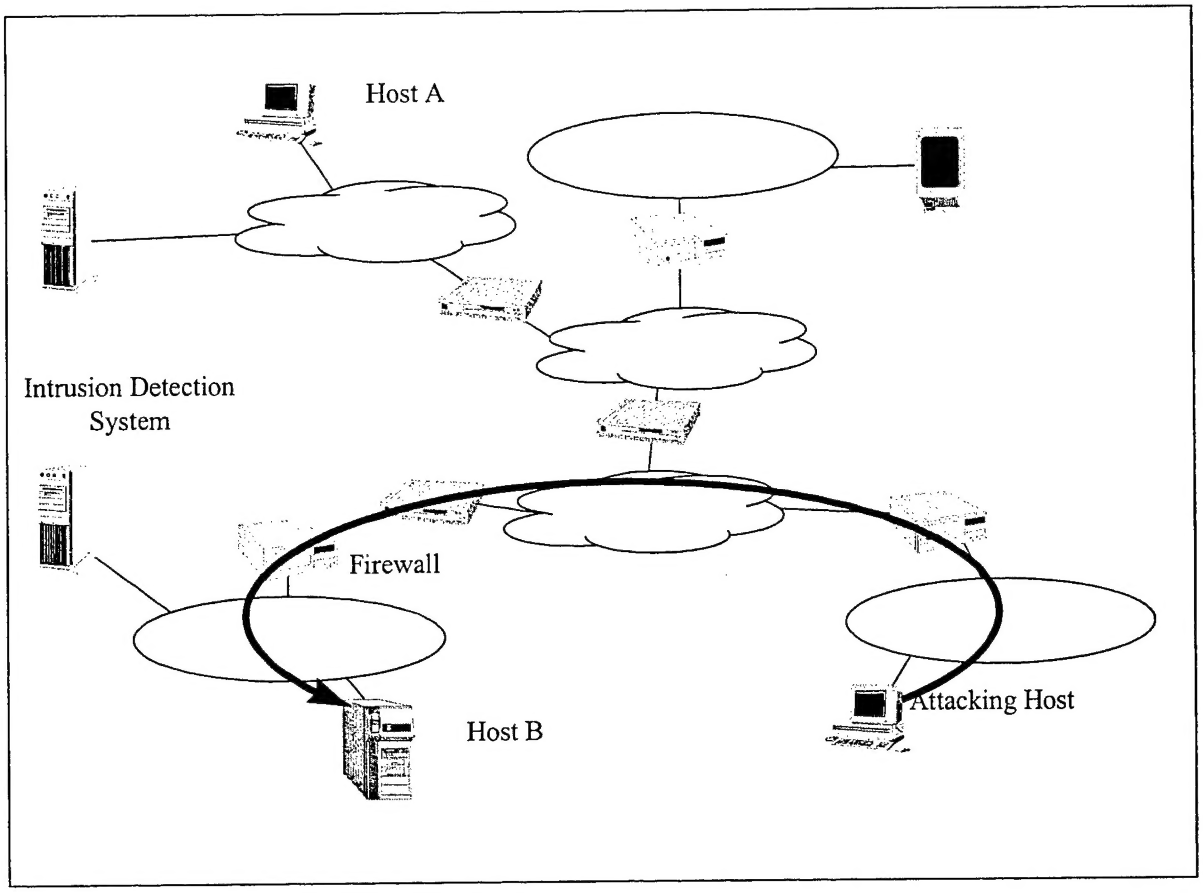

Figure 1. Spoofing Attack

1.2 Approach

The approach taken resulted in the development of a special protocol (the Intruder Detection and Isolation Protocol – IDIP) for the purpose of supporting real-time intrusion response. The protocol enables any security component that detects an attempted attack to trace the attack to its origin , to cooperate with other systems security components in isolating the attacker by blocking the attack traffic. This cooperation increases system survivability. Figure 2 shows the various components that can participate in an IDIP-based response. Intrusion detection components initiate IDIP response messages, and can support damage assessment and recovery operations within the local environment. Boundary controllers (e.g., routers and firewalls) provide network-based response mechanisms by blocking the intruder's access to network resources. Hosts would be capable of "finer-grained" responses by "killing" selected processes and connections specifically associated with intrusion attempts. A network management component (the Discovery Coordinator) receives intrusion reports and audit data from other IDIP components, enabling it to (1) provide a global picture of the intrusion events and (2) coordinate the overall system response.

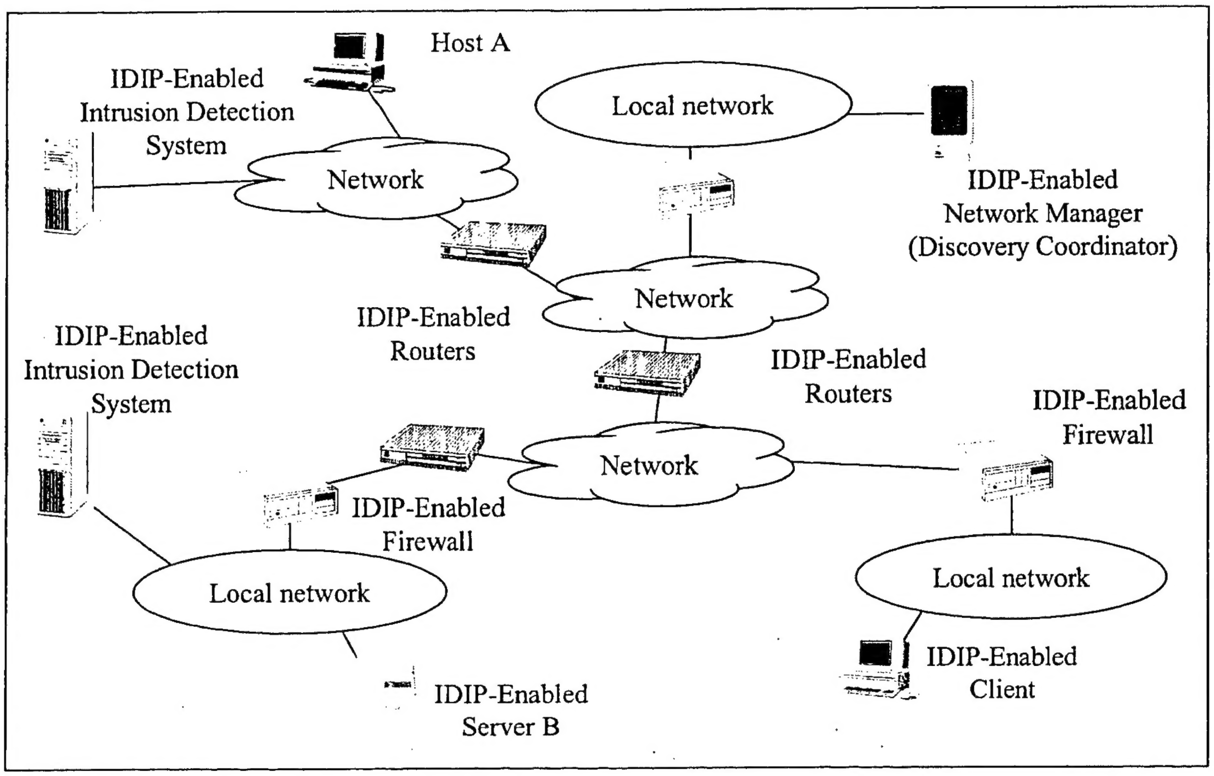

Figure 2. IDIP Components

Section 2.1 describes how these components can be used in responding to an attack.

1.3 Summary of Accomplishments

The utility of our intrusion response concept, protocol, and mechanisms were validated via two successive demonstrations. In the second proof-of-concept demonstration (see Section 6), the intrusion response system demonstrated the trace back and blocking functions for several detected attacks across a relatively complex network.

The IDIP prototype satisfies the project objectives in the following areas.

1. Scalability from small to very large systems. The use of IDIP "neighborhoods" (see Section 2) limited the knowledge required of each IDIP component enabling easy growth of the IDIP system. IDIP components only have to know about other IDIP components that are near by, plus their discovery coordinator. This reduces the management required for each component.
2. Robustness. IDIP was able to continue intrusion response operation even during attacks that flooded the network or slowed down IDIP components. Additional work could, however, further improve this robustness.

3. End-user transparency. No application changes were required to allow IDIP operate. IDIP uses minimal network resources unless the system is under attack. At that point, IDIP still uses very few network resources.
4. Simplicity. The IDIP application layer was particularly simple to implement and integrate with our component prototypes. The use of a simple User Datagram Protocol (UDP) based protocol enabled quick development of the IDIP message layer. The primary complexity in IDIP is the key management functions required to provide IDIP self protection.
5. Protection against spoofing. IDIP cryptographic services provide protection against the spoofing of IDIP components through authentication and integrity mechanisms for IDIP messages. We use a standard keyed hash algorithm to support authentication within each IDIP neighborhood.
6. Compatibility with multiple encryption technologies. Although our initial implementation uses Fortezza hardware, our cryptographic mechanisms are algorithm independent, enabling integration of additional cryptographic algorithms to support varying user requirements.

1.4 Scope

This technical report summarizes the results, including–

1. Summary of IDIP concept of operations and protocol layers.
2. Summary of project accomplishments, capabilities developed, and lessons learned.
3. Description of the cryptographic mechanisms used to support source authentication and data integrity.
4. Recommended future work to better exploit this technology, including changes to IDIP to improve it's intrusion response capabilities.
5. Description of the proof-of-concept demonstration configuration and the scenarios demonstrated.

The Intruder Detection and Isolation Protocol has evolved into a number of protocols, including IDIP message layer ([1]), HELLO protocol (for neighborhood management) ([2]), neighborhood key information distribution (NKID) ([3]), IDIP authentication header ([4] and [6]), IDIP encapsulating security payload ([5] and [7]), and IDIP application layer ([8]). These are each documented separately to enable easier document maintenance. Each of these protocols is largely independent of the others minimizing the impact of changes.

IDIP SUMMARY

There are several types of IDIP devices, including intrusion detection components, boundary controllers (i.e., firewalls and routers), network management (the Discovery Coordinator), and end systems.

For IDIP, there are three terms that require definition:

- Neighborhood – An IDIP neighborhood is a collection of adjacent IDIP components (i.e., two IDIP components are neighbors if they do not have an IDIP component between them).
- A Discovery Coordinator is an IDIP component that receives attack descriptions and descriptions of each IDIP node's response, and potentially directs the overall system response. Each IDIP node typically would have a single discovery coordinator.
- A "community" is a set of IDIP neighborhoods sharing a common Discovery Coordinator.

Figure 3 shows a typical IDIP community. Communities can either be peers or hierarchically related. This models the administrative environment in which IDIP components operate. Within a large enterprise , there may be a single discovery coordinator that serves the entire organization. Different sub-organizations within the enterprise may have their own discovery coordinators that live below the Enterprise's Discovery Coordinator in the hierarchy. Different organizations will have peer discovery coordinators. Essentially, each discovery coordinator corresponds to an administrative domain and discovery coordinators have reporting relationships that follow the relationships of the corresponding administrative domains.

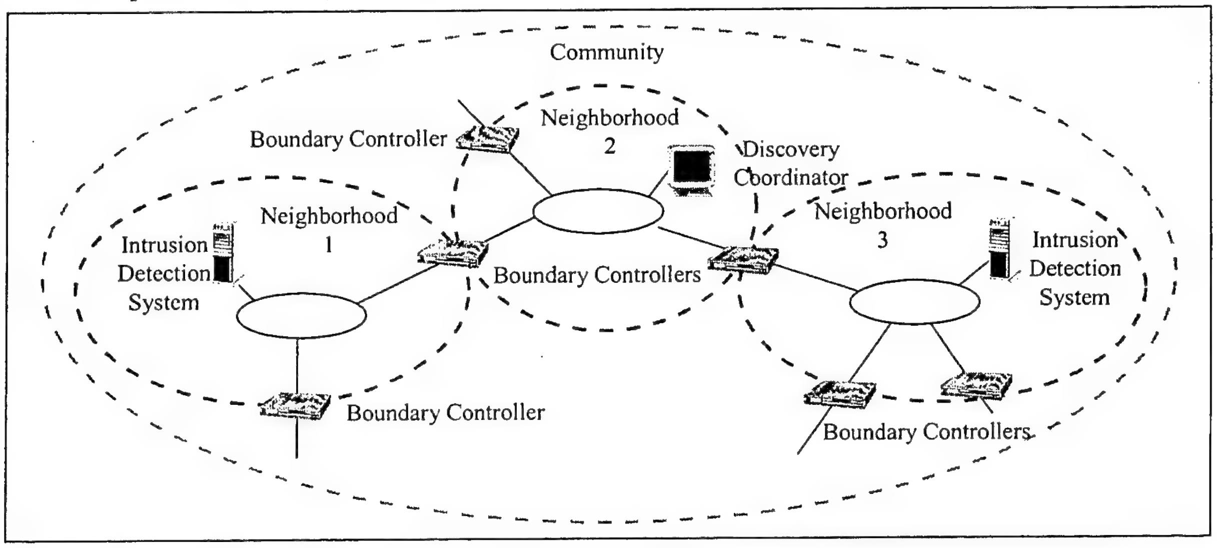

Figure 3. Typical IDIP Community

The following sections summarize IDIP concept of operations and the protocol layering of IDIP.

1.5 IDIP Concept of Operations

IDIP can be used to track intruders across intermediate networks, temporarily blocking further activity from the intruder if that activity is interfering with the system's mission. This is illustrated in Figures 3 through 5. This shows how IDIP would respond to the following attack scenario where the attacker spoofs host *A*'s address in an attack on host *B*.

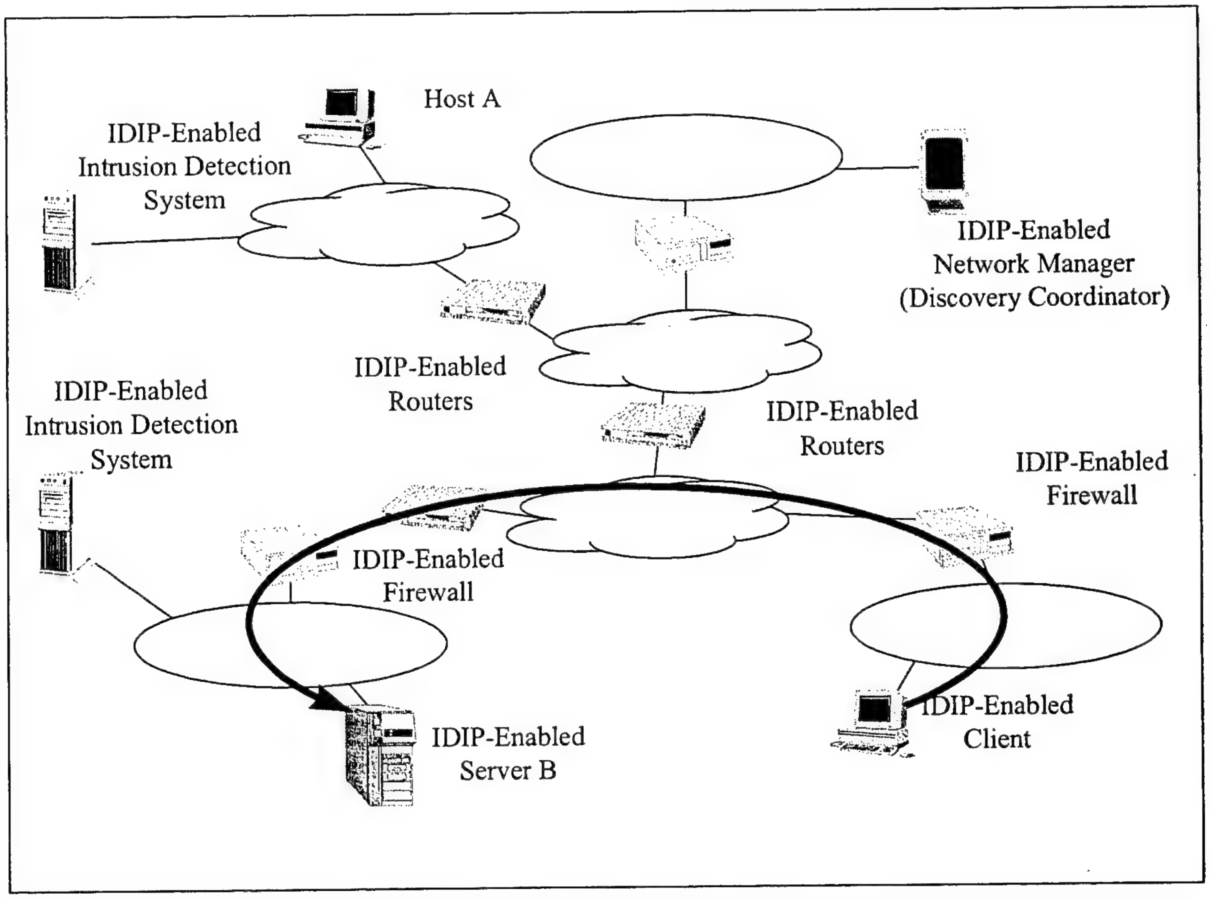
Host A
IDIP-Enabled Intrusion Detection System
IDIP-Enabled Network Manager (Discovery Coordinator)
IDIP-Enabled Routers
IDIP-Enabled Intrusion Detection System
IDIP-Enabled Routers
IDIP-Enabled Firewall
IDIP-Enabled Firewall
IDIP-Enabled Client
IDIP-Enabled Server B

Figure 4. IDIP-Based System Under Attack

Even though the attacking process spoofed *A*'s address in attacking *B*, IDIP can locate the attacker's network (Figure 5), temporarily blocking selected traffic from the attacker's packet stream if needed to protect local resources.

Once the attacker is traced back to a source machine, IDIP can be used to block traffic from that network. If the attacker's host also implements IDIP, then the IDIP boundary controller closest to the attacker can request that the source machine kill the malicious process as shown in Figure 6. Otherwise, the system administrator for the attacker's host must manually bring the host to a secure state. The temporary blocking at other IDIP boundary controllers can be lifted, and the blocking within the attacker's local network can be removed once the attacker's machine has been returned to a secure state.

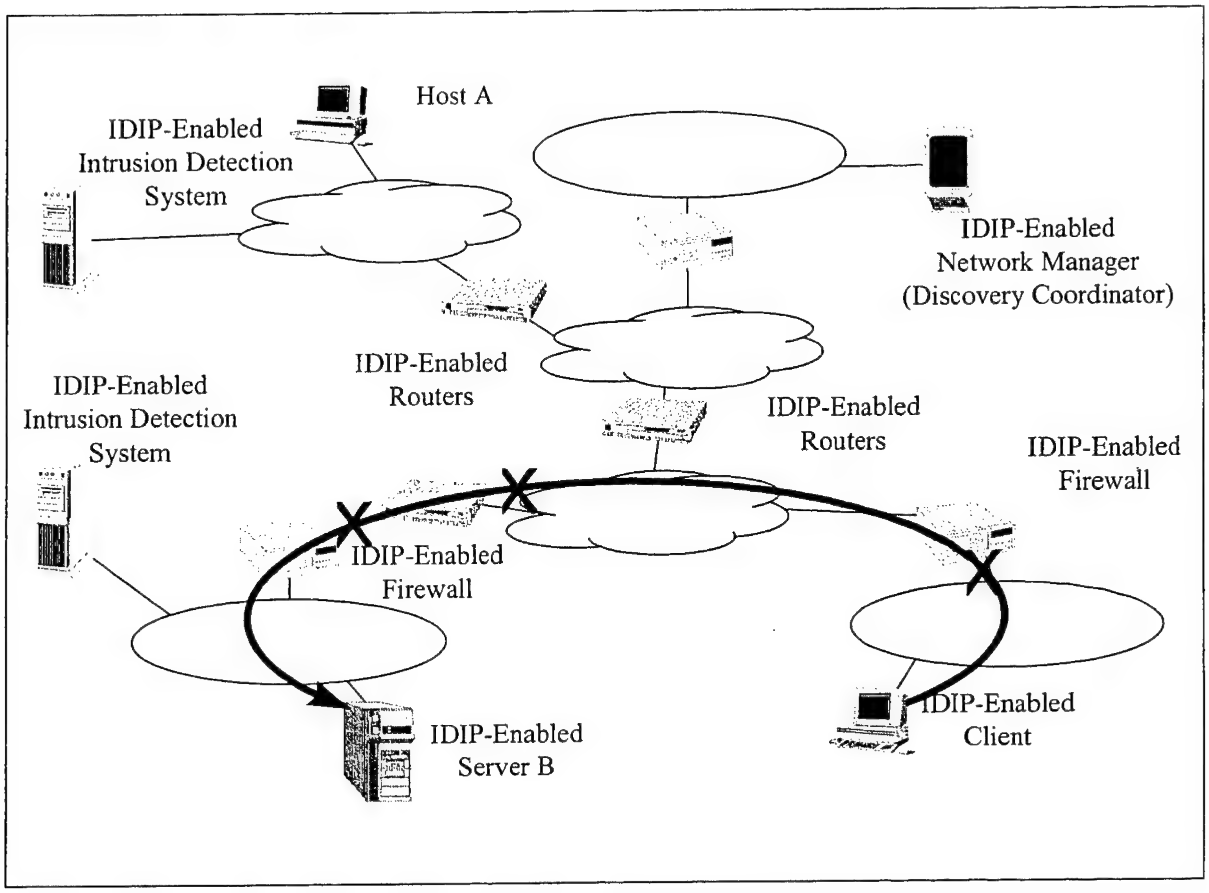
Host A
IDIP-Enabled Intrusion Detection System
IDIP-Enabled Network Manager (Discovery Coordinator)
IDIP-Enabled Routers
IDIP-Enabled Intrusion Detection System
IDIP-Enabled Routers
IDIP-Enabled Firewall
IDIP-Enabled Firewall
IDIP-Enabled Client
IDIP-Enabled Server B

Figure 5. IDIP Response

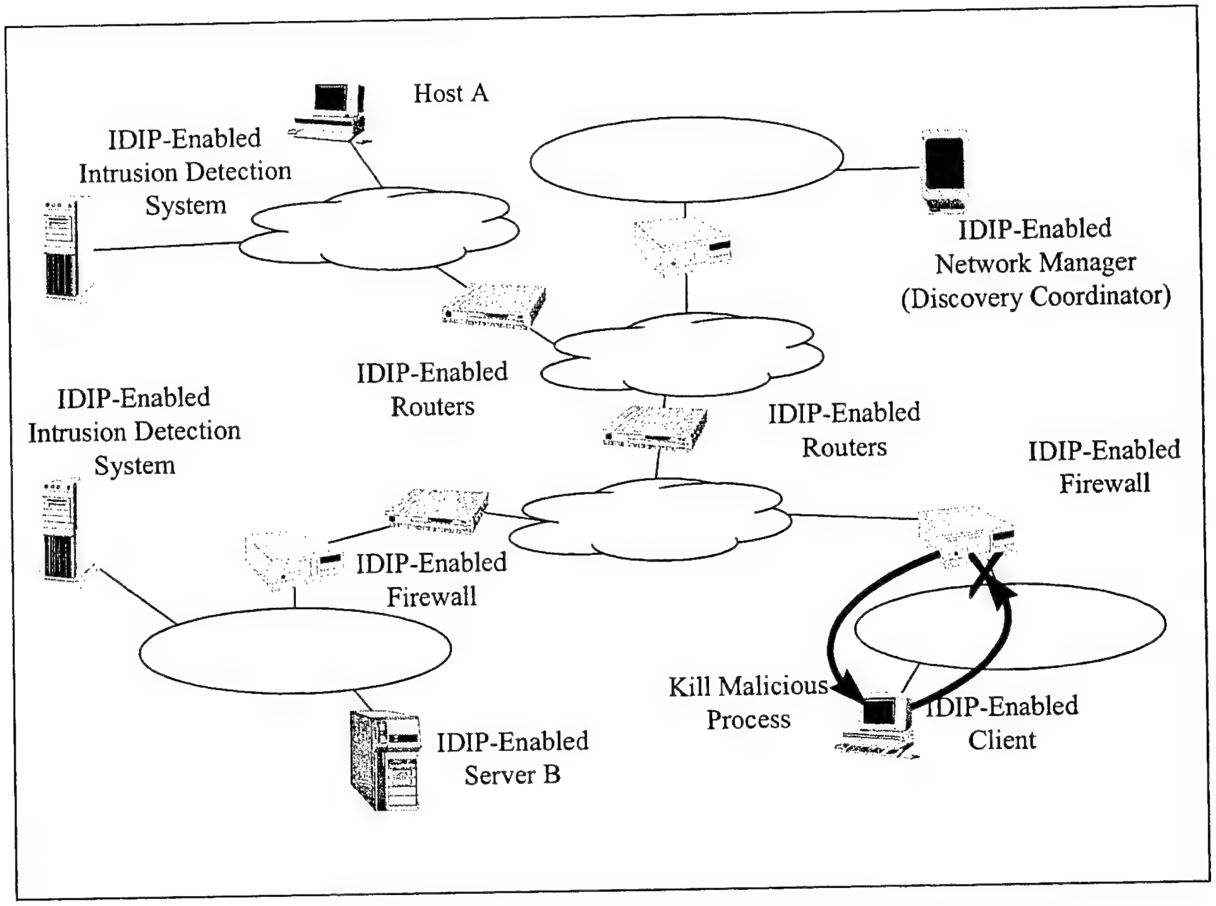

Figure 6. IDIP Response (Continued)

Throughout the response, each involved IDIP component sends response descriptions to the discovery coordinator. The discovery coordinator provides the human interface to the system, and can also request that devices modify their response.

1.6 IDIP Layering

Figure 7 shows the protocol components that comprise IDIP: (1) message layer, (2) HELLO protocol, (3) NKID, (4) cryptographic services, and (5) IDIP application layer. The following sections provide a brief summary of these protocol components. Section 4 further describes the cryptographic mechanisms.

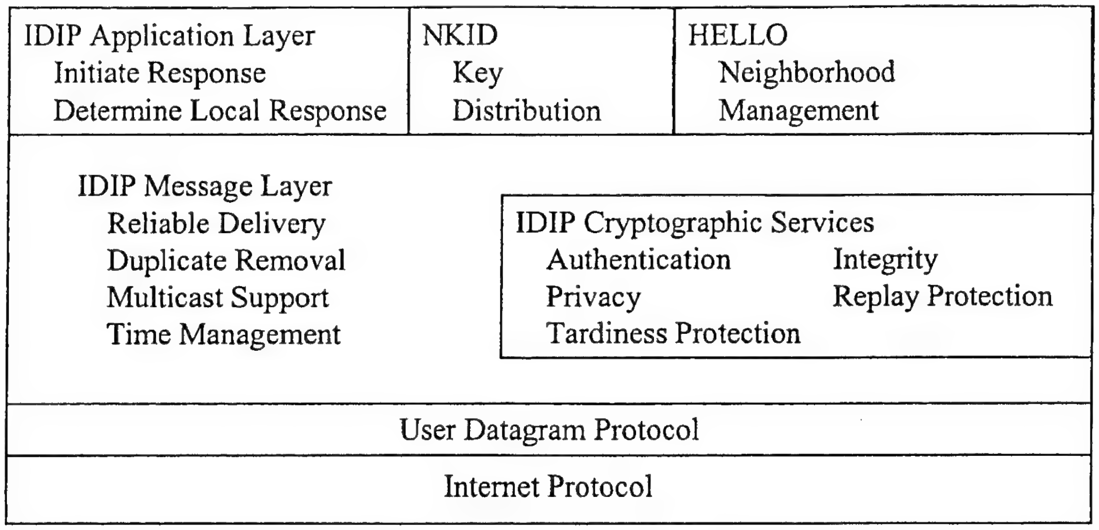

Figure 7. IDIP Layering

1.6.1 IDIP Message Layer

The IDIP message layer provides a simple, secure, reliable multicast mechanism for IDIP applications. This layer also supports uni-cast message transmission between IDIP components. The multicast functionality allows IDIP applications to communicate with all neighbors using a single Application Programmer's Interface (API) call. The message layer multicast functionality can use either the IP multicast or IP unicast service, but provides a multicast interface to the application layer.

The message layer is responsible for reliably delivering the messages to each neighbor for multicast transmission and to the single destination for unicast transmission. The message layer determines and applies the cryptographic mechanisms required for the message.

1.6.2 HELLO Protocol

The IDIP HELLO protocol provides neighborhood management functions supporting the IDIP message layer. The HELLO protocol identifies which IDIP components are neighbors and maintains the state of each neighbor so that the message layer can determine which neighbors are currently operational. The IDIP message layer provides the neighborhood management functions with notification of failed transmission, and the neighborhood management functions provide the IDIP message layer with notification of added and deleted neighbors.

1.6.3 Key Distribution

The IDIP key distribution protocol supports the cryptographic functions used by the message layer by determining a neighborhood "captain" and distributing the captain's keys throughout the neighborhood. This allows the entire neighborhood to share one key for multicast message transmission. The captain is the key generator for the entire neighborhood and is selected based primarily on a "priority" value assigned by a certification authority. This priority value indicates how good the captain is as a candidate for key generation. Criteria that may be used include likelihood of penetration and criticality to the neighborhood operation. The certification authority binds the priority and the IP address with the IDIP node's public keys in the node's credentials (which may be a standard X.509 certificate).

Captains need to be the best protected of all IDIP components because they are the components that can exclude other IDIP devices if those devices are determined to be compromised. Additionally, captains should be components that operate continually to support key distribution at all times.

1.6.4 Cryptographic Extensions to the Message Layer

IDIP provides both privacy and authentication mechanisms. These mechanisms are modeled after IP Security (IPSEC) ([10] and [11]), except that they provide protection above the transport layer. This enables IDIP support for cryptographic mechanisms in systems where the local infrastructure does not provide cryptographic protection.

1.6.5 IDIP Application Layer

The IDIP application layer protocol is responsible for initiating and responding to IDIP TRACE messages, and for responding to directives from the discovery coordinator. IDIP TRACE messages are requests for other IDIP components to respond to an anomaly. These messages include a description of the anomaly, as well as general requested responses to track down, increase monitoring for, or block the anomaly. The IDIP application layer uses the message layer to multicast TRACE messages to the appropriate neighborhoods when an anomaly is detected. When a TRACE message is received, the IDIP application layer is responsible for responding to TRACE requests by (1) searching audit information for evidence of the attack passing through the device, (2) determining the local response for a TRACE message, (3) forwarding the TRACE message to the appropriate neighborhoods after making necessary translations (address and port changes at firewalls), (4) reporting the response taken to the discovery coordinator and the initiator, and (5) sending evidence to the discovery coordinator.

2 PROJECT ACCOMPLISHMENTS

The following sections detail the major accomplishments of this project, specific capabilities developed, and lessons-learned.

2.1 Overall Accomplishments

The major achievement of this contract was the development and validation of our intrusion response approach. The current and near-term planned enhancements are consistent with our initial concept ([9]), although a number of details have changed.

Our original objective was to develop an approach to real-time intrusion response that would scale well across very large networks (i.e., thousands of network elements such as a large corporate intranet or the Internet) and support operation across administrative domains. This requires that response components be able to respond autonomously. Each component makes independent response decisions based on IDIP messages and policy parameters from the local (and parent if hierarchical system administration is used) administrative domain. The prototypes developed were used to demonstrate this over a reasonably complex network. Because of time constraints, we were unable to test the prototypes across a large operational network. However, we were able to construct a relatively complex test network (see Section 6) and the prototypes successfully operated in that network responding to a number of network-based attacks, validating the utility of our approach.

The protocol and operational concept developed meets our top-level project objectives as follows–

1. Scalability from small to very large systems.

   IDIP uses minimal system resources until anomalous behavior is detected. After initialization, only infrequent "keep-alive" messages are sent between IDIP components. IDIP uses UDP to minimize use of system resources when under attack. Our use of neighborhoods requires minimal configuration of IDIP components. Because IDIP provides all required information for response in a single message for each anomaly, component interactions are minimized. The concept allows either globally or locally set policies to be used, but once the policies are established for a component that component makes response decisions autonomously.

2. Robustness.

   The use of UDP improves system robustness. When components are failing, complex protocols tend to fail as well, where simple protocols, such as UDP, are more likely to succeed. Cryptographic authentication mechanisms provide protection of the IDIP system from attack. The use of a largely stateless protocol provides relative simplicity, which enables development of higher assurance components that are less likely to be successfully attacked.

3. End-user transparency.

   The IDIP system places no constraints or requirements on end-user operation. IDIP is compatible with the existing IP protocol suite, which enables use of IDIP without modification in most systems.

4. Simplicity.

   As stated above, the use of UDP and a relatively stateless protocol provide simplicity.

5. Protection against spoofing.

   Cryptographic authentication mechanisms are used to ensure source authenticity for IDIP participants.

6. Compatibility with multiple encryption technologies.

   Our use of the algorithm independent IPSEC-like protocol for cryptographic authentication and privacy enables use of multiple cryptographic algorithms.

Although not an original objective, we provided significant input to the DARPA Common Intrusion Detection Framework (CIDF) effort, including development of (1) the CIDF message layer and (2) initial CIDF response messages. The IDIP message layer provides a secure, reliable, light-weight mechanism for transporting CIDF messages, and is being adapted to support CIDF requirements. To enable intrusion detection components to use a single "language" for describing events, we are working with CIDF to include intrusion response-related messages in CIDF.

2.2 Capabilities Developed

The following list summarizes the work completed under this project.

1. Development of the initial IDIP and its concept of operations, documented in [9].
2. Development and integration of four prototype implementations of IDIP.
   a. **Master Intrusion Detection System (MIDS).** We developed MIDS to provide detection capabilities comparable to COTS real-time network-based intrusion detection systems. MIDS was written in Perl to allow rapid modification of MIDS to support detection of new attacks as they are developed. The MIDS architecture is shown in Figure 8. MIDS uses multiple MIDS detection agents, a central MIDS controller, and an IDIP engine. Each detection agent is a relatively simple intrusion detection component that monitors the network for one specific type or class of attacks. The detection agents report anomalies to the MIDS controller, which filters the messages for duplicates and checks thresholds where necessary before forwarding the message to the IDIP engine. The IDIP engine builds the IDIP message and sends the message to the IDIP neighbors. MIDS has proven useful in quickly building detectors for new attacks that are published on hacker bulletin boards. MIDS runs on Solaris.

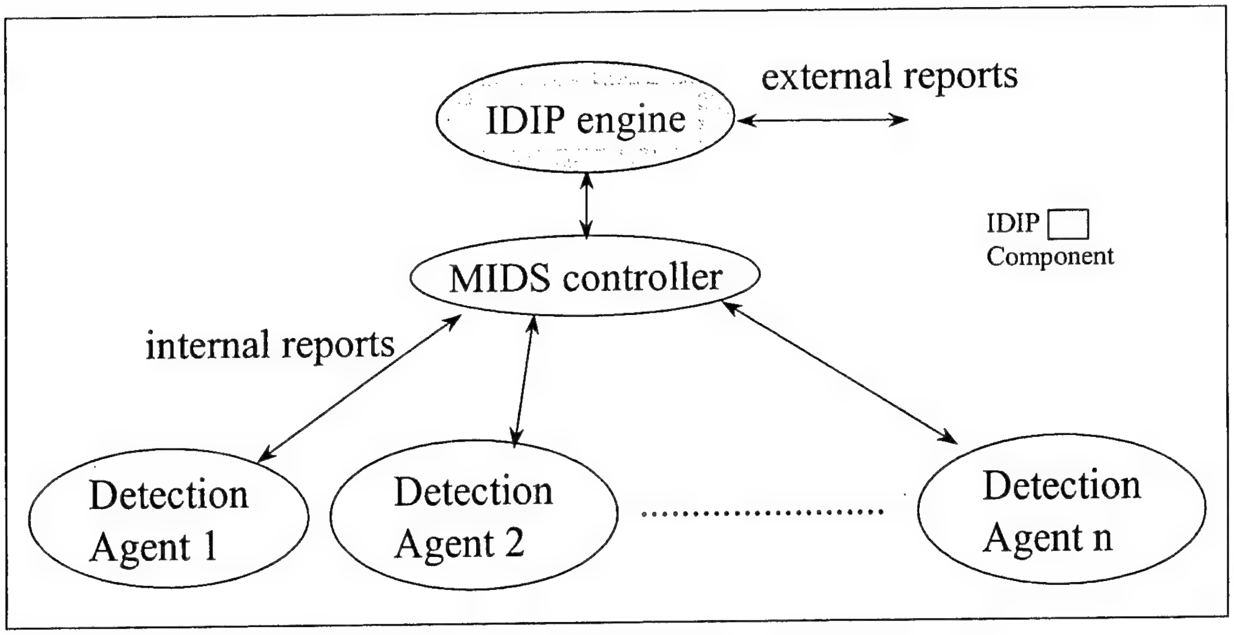

Figure 8. MIDS Architecture

b. **Firewall Toolkit.** The publicly available Firewall Toolkit (developed by Trusted Information Systems) was used to integrate IDIP with a firewall component. In the first demonstration, we used the standard Firewall Toolkit. In the second demonstration, we used the DTE-enhanced Firewall Toolkit. The DTE enhanced Firewall Toolkit uses domain-type-enforcement (DTE) technology to limit what each process, including privileged processes, in the firewall is permitted to do. These limits provide protection against flaws in the protocol stack for each service. If there is a flaw in the Firewall Toolkit or underlying operation system that an attacker can leverage, DTE places severe limitations on what the attacker is capable of doing if the attack succeeds. The attacker can only use what is provided inside the process's domain. So a compromise in one service does not result in a compromise of the entire system. The primary modifications to the Firewall Toolkit are shown in Figure 9. Changes required to the Firewall Toolkit proper were primarily enhancing the audits to enable intruder tracing. Our Firewall Toolkit implementation runs on a PC running BSDI's BSDOS 2.1. This IDIP implementation is being made available on the Firewall Toolkit web site.

c. **Secure Network Server (SNS).** The Boeing SNS was used to integrate IDIP with a filtering router. The SNS required minimal modification to support IDIP. Figure 10 shows new and modified processes required to integrate IDIP into the SNS. The largest modifications were to the SNS management process, which provided the application-layer IDIP implementation in the SNS. The SNS runs on a PC with a Boeing proprietary operating system.

d. **Discovery Coordinator.** We developed a simple discovery coordinator to support our demonstrations. The discovery coordinator is necessary to provide a single point at which one can monitor the progress of IDIP in intrusion response. The discovery coordinator software runs on a Solaris workstation, and prints summaries of messages from other IDIP components that indicate IDIP response progress.

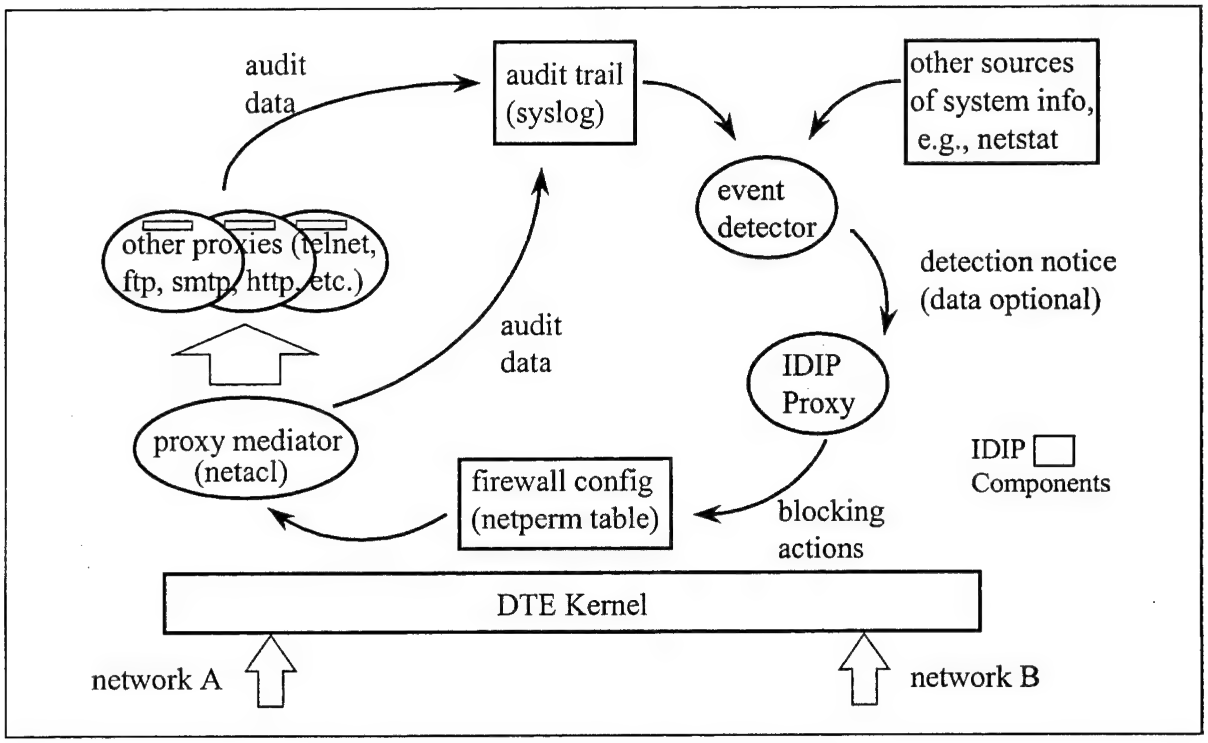
audit data
audit trail (syslog)
other sources of system info, e.g., netstat
event detector
other proxies (telnet, ftp, smtp, http, etc.)
detection notice (data optional)
audit data
IDIP Proxy
proxy mediator (netacl)
firewall config (netperm table)
blocking actions
IDIP Components
DTE Kernel
network A
network B

Figure 9. Firewall Toolkit IDIP Architecture

3. Development of cryptographic mechanisms to provide privacy, integrity, and authentication services for IDIP. These are described in sections 2.3, 2.4, and 3.3, and references [3] through [7].

4. Enhancement of the IDIP message layer to support CIDF requirements, evolving IDIP requirements, and cryptographic mechanisms. The initial IDIP message layer was substantially different from the current message layer. The current message layer better supports multicast, which when used reduces the IDIP system overhead. We have not yet experimented with multicast use, but have made transitioning to multicast operation transparent to the application layer. We also modified the partitioning of the IDIP functionality between the message and application layers primarily due to requirements levied on the message layer from our cryptographic mechanism design, but secondarily to support IDIP message layer use in CIDF. Additionally, we made several implementation changes to the IDIP message layer to accommodate the poor performance from the Fortezza cryptographic hardware. The resulting message layer protocol should be more robust and easier to maintain.

5. Development of an "attack suite" to support IDIP testing. We collected and developed a number of attacks that include several recent attacks available on hacker bulletin boards. We also developed a graphical user interface for using these attacks, which made launching these attacks easy once the scripts were developed.

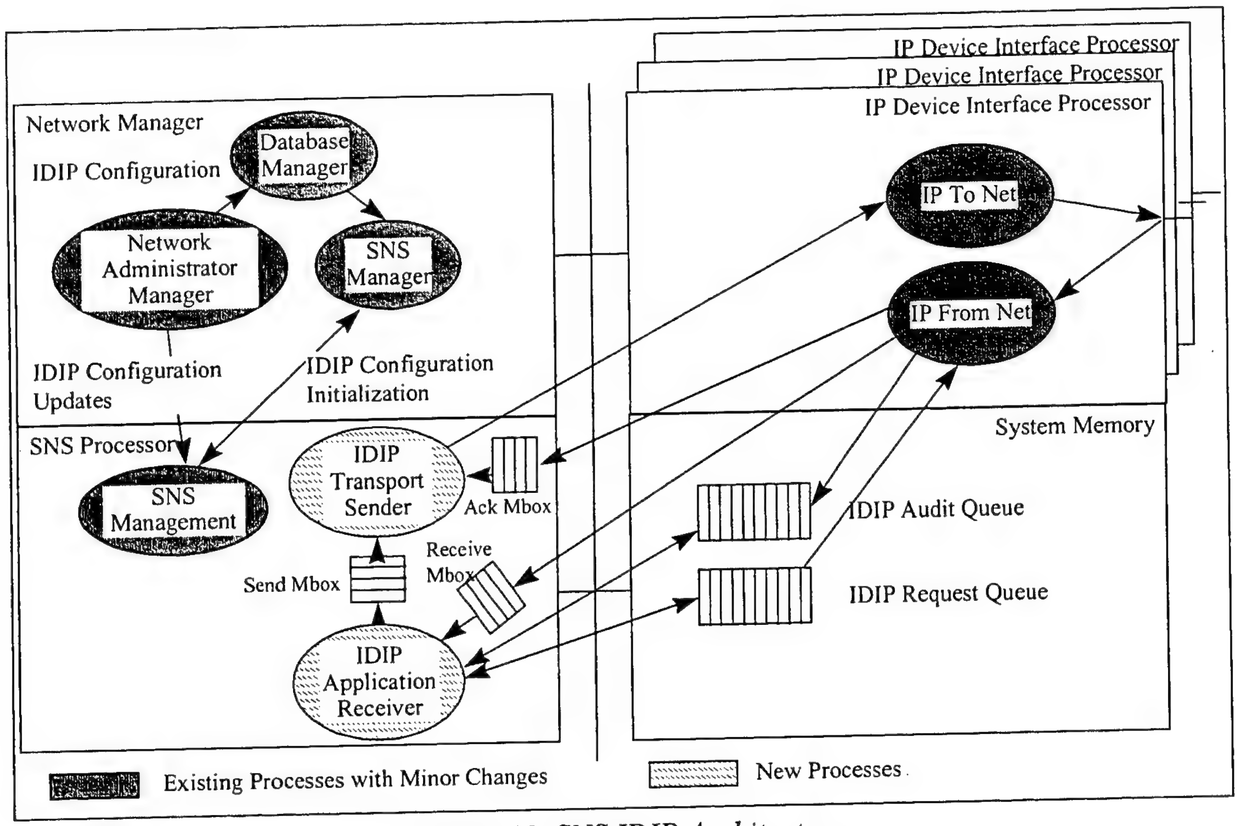
IP Device Interface Processor
IP Device Interface Processor
IP Device Interface Processor
Network Manager
IDIP Configuration
Database Manager
Network Administrator Manager
SNS Manager
IP To Net
IP From Net
IDIP Configuration Updates
IDIP Configuration Initialization
SNS Processor
SNS Management
IDIP Transport Sender
Ack Mbox
System Memory
IDIP Audit Queue
Send Mbox
Receive Mbox
IDIP Request Queue
IDIP Application Receiver
Existing Processes with Minor Changes
New Processes

Figure 10. SNS IDIP Architecture

2.3 Lessons-Learned

The following paragraphs summarize key lessons-learned from this effort.

1. **Cryptographic mechanism performance impact.** During our integration of the revised IDIP components with cryptographic mechanisms, we found that our Fortezza implementation was extremely slow. Some performance improvements were seen from using the latest release of the Cryptologic Interface (CI) library. We also noticed a significant performance difference between different Fortezza component vendors – a factor of 4 for just the card reader performance. Even with the faster drivers and hardware, overall performance appears too slow for large-scale use. With our fastest implementation, each IDIP TRACE message took approximately 100 ms for cryptographic processing on either transmission or reception. Because IDIP cryptographic mechanisms are all "hop-by-hop" (i.e., cryptographic protection is between IDIP nodes and not end-to-end) for messages that are being forwarded (e.g., messages to the discovery coordinator), each node must validate and decrypt the message on reception and encrypt and hash to forward the message. This totals 200 ms for the message, which gives an IDIP throughput of less than 5 messages per second. For messages consumed or generated by a node the cost is 100 ms, or a throughput of 10 messages per second. The IDIP boundary controllers closest to the discovery coordinator see the largest amount of traffic. In an environment such as used for our final demonstration, for various attacks, we had 7 to 15 IDIP messages for the node closest to the discovery coordinator to forward, and another 6 messages that were either consumed or generated at the node. The total

latency added by just this component for the last message to get to the discovery coordinator was over 3.5 seconds when the system was working well and both ends of the malicious connection terminated quickly. When one side of the connection would remain open (for up to 2 minutes), additional IDIP messages would be sent for each additional application packet that the malicious connection was generating. Each of these would further add another 3.5 seconds of latency, backing up the IDIP components. The result was that the IDIP components would lose connectivity with each other. On loss of connectivity, the components were designed to attempt to regain communication and then proceed through key exchange. Key exchange requires public key cryptography, a slower operation, which further aggravated the problem. We made several changes to the software architecture, the protocol definition, and the implementation to (1) reduce the number of IDIP messages and cryptographic operations required; (2) make the message layer more tolerant of slow nodes; and (3) reduce the impact of cryptographic mechanisms on the performance of the HELLO protocol, which is used to maintain connectivity. A side benefit of this slowness was that the resulting message layer implementation was more robust than previous versions.

2. **Cryptographic mechanism integration.** We did not allocate enough time and resources to the cryptographic mechanisms. Our initial approach was to use IPSEC, and adapt it to use Fortezza hardware. However, IPSEC was not readily available and not as easily modified as we had hoped. In addition, we concluded that application-layer encryption would better suit our purposes by enabling IDIP use where IPSEC is not deployed and enabling application-layer authentication, rather than node-level authentication. Because we were forced to develop our own cryptographic mechanisms, this task took significantly longer than expected. We also found that debugging cryptographically protected network applications is much more difficult (snoop is ineffective) than standard applications. This increased the time required for the final system integration. The poor Fortezza performance also contributed to additional integration problems, which required more time than expected for trouble-shooting. Finally, development of a key distribution protocol proved more difficult than expected, requiring additional integration time.

3. **Shared message layer.** Developing a common message layer to be shared by the three development teams reduced integration time. We were able to integrate IDIP with MIDS in one day, and integrate all four components with each other in less than a week in preparation for our first demonstration. For the second demonstration, once the cryptographic mechanisms were integrated with the message layer, component integration took very little time.

4. **Independent development.** Although we were initially concerned about independent development with a short integration time-frame for the four components, this did not result in any problems. Splitting program responsibilities by component enabled the three development groups to work largely independently once the concept definition phase was completed. This was further supported by the early documentation of the concept and protocol details.

5. **Cryptographic algorithm independence.** Defining an algorithm-independent cryptographic protocol enables us to replace the poor performing Fortezza implementation with higher speed components without disturbing the system protocol

architecture. We also partitioned the implementation so that the Fortezza-specific software is encapsulated in a single module. We anticipate a number of software changes as we add the second set of cryptographic algorithms, but that should be minimal except for the algorithm-specific module.

6. **Loosely coupled messaging architecture.** The use of a simple message-based architecture eased the integration efforts and allowed easy evolution as we changed various parts of the protocol. Each message carries all of the state information required for a component to determine the correct response, which eliminated inter-component dependencies. This architecture also reduced the timing dependencies to those that exist at the message layer.

7. **Protocol independence.** We recognized after the initial implementation that the infrastructure components could be split into multiple separable protocol components. We have documented each separately (references [1] through [8]) to promote maintaining this separation and independence between the different parts of IDIP. The implementation still requires some work to complete the separation.

3 CRYPTOGRAPHIC MECHANISMS

We developed cryptographic privacy, authentication, and integrity mechanisms, modeled after the IPSEC mechanisms. The top-level goals for our IDIP cryptographic mechanisms are–

1. Intra-neighborhood (i.e., hop-by-hop) integrity/authentication and confidentiality protection. In communicating with a non-neighbor (e.g., discovery coordinator), IDIP protects the message between adjacent IDIP neighbors on the path to the target. At each IDIP hop, the message is validated and decrypted, then encrypted and hashed for the next hop. No end-to-end privacy or integrity mechanisms are provided at this time.
2. Single neighborhood keying relationships. Each neighborhood has a single key for privacy and another key for the integrity/authentication mechanism. Because there is a single key, all messages are authenticated to have come from a neighbor, but the specific neighbor identity is not authenticated. Also, each neighbor can decrypt any message sent within the neighborhood. This was done to minimize the number of keys required to support each neighborhood, which reduces the performance impact of key management and simplifies the implementation. This requires that a single node generate the neighborhood key. This node is called the neighborhood "captain," and is selected based on management-specified attributes.
3. Separate key management and data protection. The NKID protocol provides key management functions through a separate mechanism from the application data. That is, the keying information is not sent as part of the application data, but is established prior to application data being transmitted. This enables us to avoid performing key management functions during real-time intrusion response. NKID supports periodic key change based on either time or specific events (e.g., neighbor compromise).
4. Algorithm independent data protection protocol. The IDIP cryptographic mechanisms allow different neighborhoods to use different cryptographic algorithms. The initial implementation uses the Fortezza supported algorithms: Key Exchange Algorithm (KEA) for key exchange, DSS for signatures, Skipjack for privacy, and HMAC-SHA for authentication/integrity, however, the protocol can support alternatives to these. The algorithm set is selected by the neighborhood captain.

In addition to these goals, we placed the following requirements on these mechanisms–

1. Support for multicast operation, where a neighborhood is the multicast group. Using a common neighborhood key enables the message layer to apply the cryptographic mechanisms to a multicast message and have that message readable by all recipients in the neighborhood.
2. Simplicity. The use of IPSEC-like headers for authentication and encryption provides all the state information required with each message, except for the secret keys distributed through NKID. The implementation of the privacy and authentication protection mechanisms proved simple, with the only complexity in the key management software.
3. Operation over connectionless (UDP) channels. The protocol was specifically designed to operate over UDP. The cryptographic association is tied only to neighborhood membership and not to protocol state.

4. Minimal performance impact during real-time operation. The IPSEC-like headers used support the cryptographic mechanisms have minimal size. In addition, because these headers carry the necessary state information, there is no interruption of real-time operation for association negotiation or resynchronization.
5. Support for tardy delivery detection. The message layer header includes a transmission timestamp, and because the header is covered by the integrity mechanism, this field can be used to detect delayed delivery. This is not a fool-proof mechanism because we do not modify messages being retransmitted (to avoid additional cryptographic processing), so retransmitted messages have the same timestamp as the original and messages dropped due to transmission error also appear tardy.
6. Detection of modification to either IDIP application data and message header. The application of HMAC-SHA algorithm to the entire IDIP message enables detection of modification during transmission.
7. Protection against message replay. The message layer header includes a unique sequence number for each message sent, which along with the application integrity mechanism to the header, enables detection of message replay.
8. Minimal dependencies on and interactions with the other IDIP mechanisms. The cryptographic services are called from the message layer. The security associations for the cryptographic mechanisms are established by NKID, but NKID could be easily replaced by any other protocol that exchanges keys for multicast groups. NKID uses the message layer, but could use any other protocol that support uni-cast operation. NKID only relies on the HELLO protocol for notification of neighborhood changes so that it can determine when key exchange is required. The IDIP message layer relies on the cryptographic services to be sufficiently fast so that use of the cryptographic mechanisms does not cause messages to time-out and be retransmitted. (This last dependence was where most of our cryptographic mechanism integration problems arose.)

4 FURTHER INVESTIGATIONS, RESEARCH, AND DEVELOPMENT

The following sections describe (1) changes to IDIP that would provide more capable intrusion response and (2) additional research and development to gain additional experience with intrusion response mechanisms.

4.1 IDIP Modifications

We have already begun identification of IDIP modifications to enable more adaptable, optimal responses under a related effort (Contract F30602-97-C-0217). Those modifications are discussed in more detail in [12].

4.2 Further Research and Development

Beyond these changes, the following additional work would improve IDIP functionality in operational environments–

1. Analysis of IDIP message layer robustness, and improvements, where needed, to improve survivability in the face of infrastructure attacks.
2. Additional support for CIDF emerging requirements. CIDF may require transmission of very large objects, requiring extensions to the IDIP message layer to support multi-protocol data unit messages. Other extensions include directory support and option negotiations.
3. Analysis of the HELLO protocol to identify additional mechanisms to support more dynamic neighborhood building.
4. Analysis of NKID robustness. Specifically, identification of methods to reduce the risk of key distribution during intrusion response.
5. Investigation of alternate protocols for NKID, HELLO, and IDIP message layer. Since our initial design for these infrastructure components, a number of proposals have been developed and submitted to IETF for reliable multicast, secure multicast, and multicast group formation. We should analyze these to determine if one of these alternatives will meet our requirements, and if not, determine if these proposals have concepts that could be folded into IDIP to improve our functionality.
6. Integration with CIDF application-layer protocol. CIDF application-layer formats for response components are just beginning to be developed. We should ensure that these support our requirements, and then transition to CIDF to enable easy integration with other DARPA intrusion detection research projects.
7. Integration of additional cryptographic algorithms. The IDIP implementation would benefit greatly from integration of higher performance cryptographic algorithm implementations. IDIP would be more robust if it could forward messages through an IDIP component with a throughput of at least 200 messages per second. Our current Fortezza implementation can only achieve 5 messages per second.
8. IDIP message layer implementation improvements. The initial IDIP message layer was written as a single protocol. We have since partitioned the protocol definitions to make

HELLO, message layer, NKID, and cryptographic mechanisms largely independent. The IDIP implementation would benefit from a restructuring to remove the remaining implementation dependencies. This would further enable independent development of each of the components. Related to this, it would be beneficial to investigate standard cryptographic APIs to determine which, if any, could be used to isolate the cryptographic mechanisms. For the initial implementation, we used the Fortezza CI API because it minimized the software development effort. However, this API is not used for other implementations.

9. Implementation of cost models within each IDIP component to allow cost-benefit determination in selecting response options.
10. Investigation of better languages (and taxonomies) to describe attacks and intrusion response actions.
11. Development of techniques for detecting and responding at the system level to cooperating, distributed attacks (e.g., worms).
12. Integration with additional components, including COTS products, to gain better operational experience with IDIP.

5 RESULTS DERIVED FROM THE PROOF-OF-CONCEPT DEMONSTRATION

Figure 11 shows the configuration used in the proof-of-concept demonstration. The objectives of the demonstration were to validate that IDIP can operate within a reasonably large network environment and to demonstrate the IDIP cryptographic protection mechanisms. We had planned to separate the halves to the demonstration configuration using the Boeing intranet, however, the extended integration time required for the cryptographic mechanisms made that infeasible. Instead, we used a Solaris workstation between the two "intranet networks" to simulate this network separation. This provided a reasonably complex network for testing IDIP. During the testing, we found a number of implementation errors, but no flaws in the basic concepts.

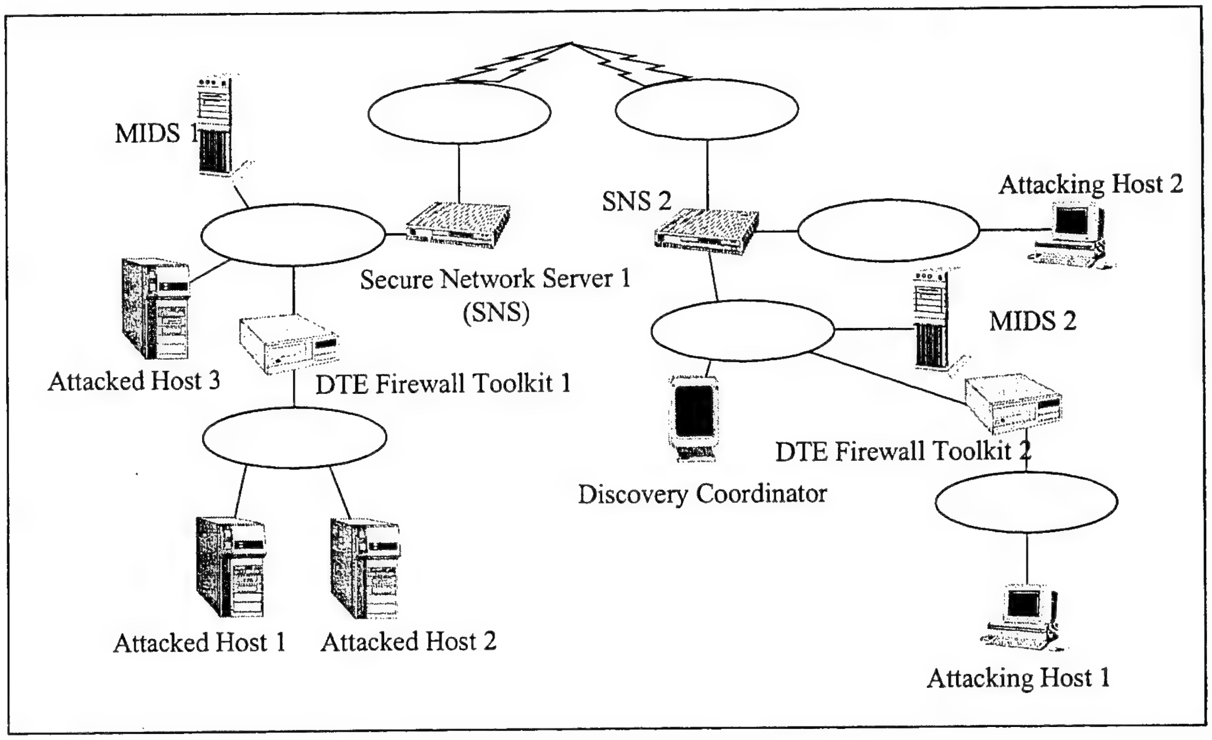

Figure 11. Demonstration Configuration

The following attacks were demonstrated, along with the IDIP response to those attacks–

1. Password Guessing attack was launched from Attacking Host 1 to Attacked Host 1 and was detected by both MIDS devices. There are two different types of password guessing attacks detected by MIDS: general password guessing (e.g., dictionary attack) and attempting well-known passwords for standard accounts (called doorknob rattling). On detection of either too many failed password attempts or access to well-known accounts, MIDS initiates IDIP response. Both MIDS devices initiated tracing and blocking. Both Firewall Toolkits closed the connection and the SNS's both blocked packets for the connection.

2. Honey pot Entry attack was launched from Attacking Host 1 to Attacked Host 1 and was detected by both MIDS devices. MIDS monitors for retrieval of directory listings with a set of file names that are not permitted to be retrieved. Both MIDS devices initiated

tracing and blocking. Both Firewall Toolkits closed the connection and the Sis's both blocked packets for the connection.

3. Unusual Connection attack was launched from Attacking Host 1 to Attacked Host 2 and was detected by Firewall Toolkit 1. The Firewall Toolkit monitors connection requests for an administratively defined list of prohibited connections, and on detection of a prohibited connection, initiates IDIP response. Firewall Toolkit 1 killed the connection and initiated tracing and blocking. Both Sis's blocked packets for the connection, and Firewall Toolkit 2 killed its end of the connection.

4. Unusual Connection attack was launched from Attacking Host 2 to Attacked Host 3 and was detected by SNS 1. The SNS packet filtering function rejects packets for any packet matching administrator specified criteria, and on packet rejection, initiates IDIP response. SNS 1 initiated tracing and blocking. Both Sis's blocked further packets being transmitted between Attacking Host 2 and Attacked Host 3.

5. Honey pot Entry attack was launched from Attacking Host 1 to Attacked Host 3 and was detected by Firewall Toolkit 1. The Firewall Toolkit monitors FTP connections for user retrieval of an administratively defined list of prohibited files (e.g., /etc/passwd), and on detection of a prohibited file retrieval, initiates IDIP response. Firewall Toolkit 1 killed the connection and initiated tracing and blocking. Both Sis's blocked packets for the connection, and Firewall Toolkit 2 killed its end of the connection.

6. Each of the following attacks were launched from Attacking Host 2 to Attacked Host 3 and were detected by MIDS 1. MIDS 1 initiated tracing and blocking. Both Sis's blocked the specified packets, however, a bug in the SNS software caused SNS 2 to not forward the blocking and tracing request to its neighbors.

   a. LAND attack. The LAND attack is sending any IP datagram with equal source and destination addresses. This causes many hosts (including NT and BSD Unix) to crash.

   b. TCP/UDP Port Scan attack. TCP Port Scan is not necessarily an attack in itself, but is often a prelude to attack. The objective of a port scan is to identify available vulnerable services on the remote machine.

   d. TCP Sequence Number Guessing attack. TCP Sequence Number Guessing involves attempting to correctly guess the next TCP sequence number in an effort to hijack a connection.

   e. TCP SYN Flood attack. TCP SYN Flood attacks attempt to consume all pending connections on the targeted machine, making it unavailable for use by other network users.

The demonstration successfully demonstrated the feasibility of using IDIP over a complex network in spite of a few minor implementation bugs. Additionally we demonstrated the use of the cryptographic protection mechanisms for providing security for IDIP.

6 SUMMARY AND CONCLUSION

This project demonstrated the feasibility of using a protocol-based solution for coordinated real-time response to network intrusions. The first-generation IDIP provides a response protocol that allows loosely coupled, independently managed security devices to cooperate in tracking down intruders and blocking their network activity, while allowing host devices to terminate malicious processes associated with that activity. While the initial IDIP provides good response capabilities, a more sophisticated capability can be achieved by extensions to the IDIP design and an upgrade to the present IDIP implementations.

6.1 Recommended Future Work

There are several areas where IDIP concept requires additional work for it to become more responsive to the demands of a wide range of environments prevalent in networking applications. Section 5 above identifies recommended changes to the protocol design and implementation. We also believe that more effort is required to use IDIP across network administrative domain boundaries and to make each IDIP component more intelligent in selecting responses that minimize collateral damage to the protected system. We further recommend that IDIP should be implemented for additional components in order to allow testing alternate IDIP response mechanisms[1]. Also, we recommend that IDIP analysis, testing, and experimentation be done to determine what changes are necessary to use IDIP on a large scale. Finally, although we believe that the discovery coordinator is a good choice for a global intrusion detection system; and we are investigating how we can use this information to detect and respond to these types of attacks; this is a difficult research area and should be pursued by multiple investigators employing a variety of approaches to determine the best solution.

6.2 Conclusion

We defined an initial intrusion response concept, a framework and a protocol within which that concept can be tested and refined. We developed four implementations of the protocol that provide good examples of the use of IDIP. We determined that development and integration of IDIP applications with various components (firewalls, filtering routers, and intrusion detection systems) is straight-forward, however, development of the infrastructure to support IDIP was much more difficult than initially planned. We have identified several potential directions in which the IDIP concept can be expanded to (1) improve its utility in very large, complex networks (i.e., thousands of network elements) and (2) more optimally respond to network-based attacks.

[1] These recommendations are being investigated and implemented under two related contractual efforts: Adaptive System Security Policies (Contract F30602-97-C-0217) and Automatic Response to Intrusion (Contract F30602-97-C-0309).

7 REFERENCES

[1] The Boeing Company. *Intruder Detection Isolation Protocol (IDIP) Message Layer (Draft)*, Boeing Document Number D658-10817-1, February 1998.

[2] The Boeing Company. *Intruder Detection Isolation Protocol (IDIP) Neighborhood Management (Draft)*, Boeing Document Number D658-10819-1, February 1998.

[3] The Boeing Company. *Neighborhood Key Information Distribution (NKID) Protocol (Draft)*, Boeing Document Number D658-10818-1, February 1998.

[4] Trusted Information Systems, Inc. *Intruder Detection Isolation Protocol (IDIP) Authentication Header (AH)*, TIS Report Number 0699D, November 1997.

[5] Trusted Information Systems, Inc., *Intruder Detection Isolation Protocol (IDIP) Encapsulating Security Payload (ESP)*, TIS Report Number 0698D, November 1997.

[6] Trusted Information Systems, Inc., *IDIP AH with Hashed Message Authentication Codes (HMAC)-SHA-1,* TIS Report Number 0700D, November 1997.

[7] Trusted Information Systems, Inc., *IDIP ESP with SKIPJACK Cipher Block Chaining (CBC),* TIS Report Number 0701D, November 1997.

[8] The Boeing Company. *Intruder Detection Isolation Protocol (IDIP) Application Layer (Draft)*, Boeing Document Number D658-108120-1, February 1998.

[9] The Boeing Company. *Protocol Definition - Intruder Detection and Isolation Protocol Concept*, Boeing Document Number D658-10732-1, January 1997.

[10] Network Working Group, *Security Architecture for the Internet Protocol*, Internet Draft, draft-ietf-ipsec-aarch-sec-01, Randall Atkinson, Cisco Systems, 10 November 1996.

[11] Network Working Group, *IP Encapsulating Security Payload (ESP)*, Internet Draft, draft-ietf-ipsec-esp-v2-00, Stephen Kent, BBN Corporation, 21 July 1997.

[12] The Boeing Company. *Adaptive System Security Policies, Preliminary Assessment*, Boeing Document Number D658-10821-1, February 1998.

GLOSSARY OF TERMS

API	application programmer's interface
ACK	acknowledgment
COTS	commercial off-the-shelf
IDIP	Intruder Detection and Isolation Protocol
IP	Internet Protocol
IPSEC	IP Security
SYN	TCP's synchronization flag
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)

The advancement and application of Information Systems Science and Technology to meet Air Force unique requirements for Information Dominance and its transition to aerospace systems to meet Air Force needs.